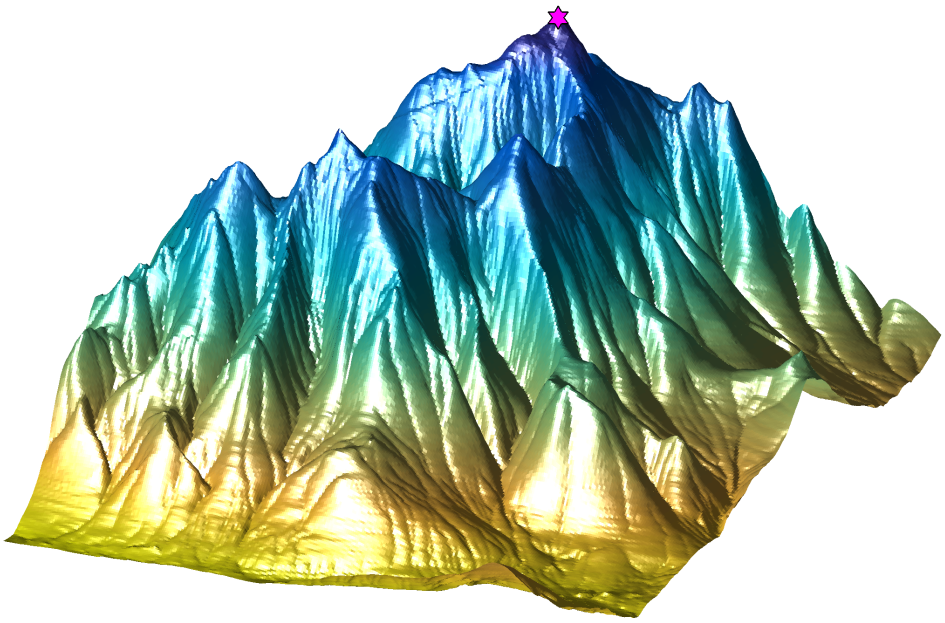
Multivariable Grid Search Method

for optimization, control and auto-calibration

September 25, 2016



Contents

1 Introduction

The Grid Search Method [1] is an optimization routine that calculates the minimum point of a multi-variable function. This method is based on a grid defined by multiple dimensions. Each dimension has a range of values. Each range is divided into a set of equal-value intervals. The multi-dimensional grid has a centroid which locates the optimum point. The search involves multiple passes. In each pass, the method updates the node (point of intersection) with the least function value. This node becomes the new centroid and builds a smaller grid around it. Successive passes end up shrinking the multidimensional grid around the optimum.

In what follows, the implementation of the Grid Search Method is presented, both as a Matlab m file and in Matlab/Simulink.

2 Matlab m file

The Matlab function is defined as:

[X, BestF, Iters] = GridSearch(N, XLo, XHi, NumDiv, MinDeltaX, Eps_Fx, MaxIter, myFx)

The function has the following input paramenters:

```
N            - number of optimization variables
XLo          - vector of lower values
XHi          - vector of higher values
NumDiv       - vector of number of divisions along each dimension
MinDeltaX    - vector of termination tolerance for each variable
Eps_Fx       - termination tolerance for the function value
MaxIter      - maximum number of iterations
myFx         - name of the optimized function
```

The function generates the following output:

```
X            - vector of optimized variables
BestF        - function value at optimum
Iters        - number of iterations
```

The GridSearch.m file is listed in what follows:

```

function [XBest,BestF,Iters]=GridSearch(N, XLo, XHi, NumDiv, ...
MinDeltaX, Eps_Fx, MaxIter, myFx)
% Function performs multivariate optimization using the
% grid search.
%
% Input
%
% N              - number of optimization variables
% XLo            - vector of lower values
% XHi            - vector of higher values
% NumDiv         - vector of number of divisions along each dimension
% MinDeltaX      - vector of termination tolerance for each variable
% Eps_Fx         - termination tolerance for the function value
% MaxIter        - maximum number of iterations
% myFx           - name of the optimized function
%
```

```
% Output
%
% X               - vector of optimized variables
% BestF           - function value at optimum
% Iters           - number of iterations
%
XHi_ev = XHi;
XLo_ev = XLo;
Xcenter = (XHi + XLo) / 2;
XBest = Xcenter;
DeltaX = (XHi - XLo) ./ NumDiv;
BestF = feval(myFx, XBest, N);
if BestF >= 0
  LastBestF = BestF + 100;
else
  LastBestF = 100 - BestF;
end
X = XLo; % initial search value

Iters = 1;
bGoOn = 1;

while (bGoOn > 0) && (abs(BestF - LastBestF) > Eps_Fx) && (Iters <= MaxIter)

  bGoOn2 = 1;

  while bGoOn2 > 0

    Iters = Iters + 1;

    X_ev = max(XLo_ev,min(XHi_ev,X));
    F = feval(myFx, X_ev, N);
    if F < BestF-Eps_Fx/10  % updated only if F changed with more than Eps_Fx/10
      LastBestF = BestF;
      BestF = F;
      XBest = X_ev;
    end

    % search next grid node
    for i = 1:N
      if X(i) >= XHi(i)
        if i < N
          X(i) = XLo(i);
        else
          bGoOn2 = 0;
          break
        end
      else
        X(i) = X(i) + DeltaX(i);
        break
      end
    end

  end % while bGoOn2 > 0


  XCenter = XBest;
  DeltaX = DeltaX ./ NumDiv;
  XLo = XCenter - DeltaX .* NumDiv / 2;
  XHi = XCenter + DeltaX .* NumDiv / 2;
  X = XLo; % set initial X

  bGoOn = 0;
  for i=1:N
    if DeltaX(i) > MinDeltaX(i)
      bGoOn = 1;
    end
  end

end % while bGoOn > 0 && () && ()
```

Example 1

First, we test the Grid Search Method on the Matlab *peaks* function, given by:

$$J(x_1, x_2) = 3(1 - x_1)^2 e^{-x_1^2-(x_2+1)^2} - 10\left(\frac{x_1}{5} - x_1^3 - x_2^5\right) e^{-x_1^2-x_2^2} - \frac{1}{3}e^{-(x_1+1)^2-x_2^2} \tag{1}$$

Function call:

[XBest,BestF,Iters] = GridSearch(2, [-3 -3], [3 3], [4 4], [1e-5 1e-5], 1e-3, 1000, ’fx1’)

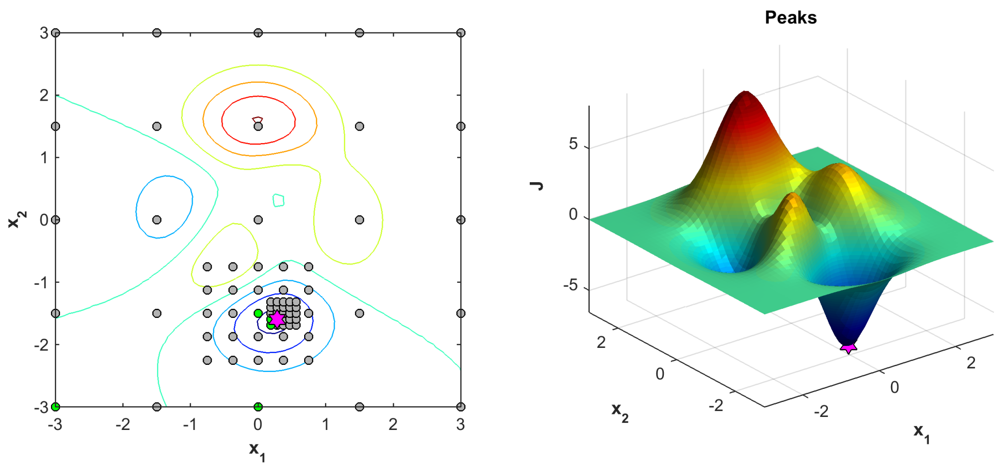


Figure 1: The Grid Search Method applied on the Matlab *peaks* function: ● - evaluation grid, ● - centroid, ⋆ - optimum.

```
XBest =

    0.2813   -1.5938

BestF =

   -6.5158

Iters =

    76

Elapsed time is 0.006446 seconds.
```

Example 2

Second, we show how the Grid Search Method can be used to find the peak of the White Mountain. The White Mountain map resides in Matlab under *mtWashington*.

Function call:

[XBest,BestF,Iters] = GridSearch(2, [310965 4902495], [320115 4915455], [5 5], [1 1], 1, 1000, 'fx2')

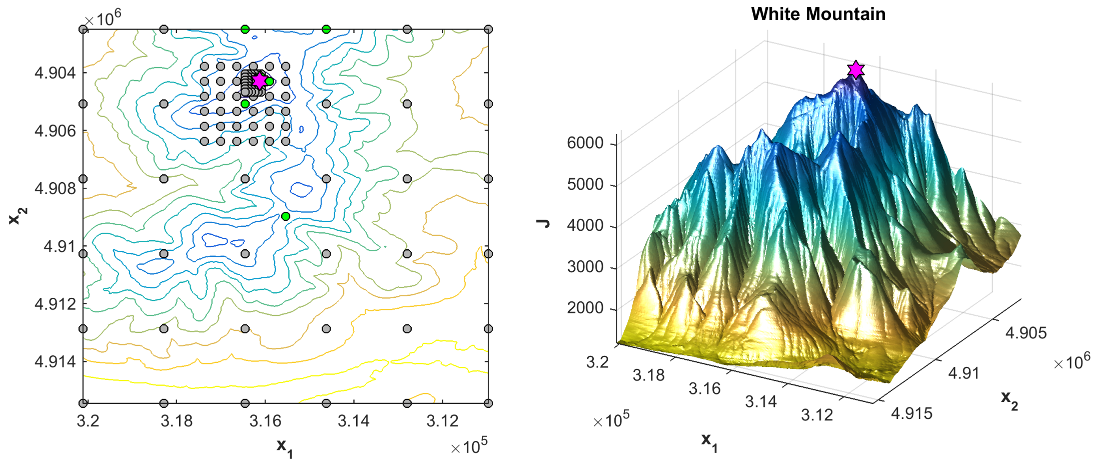


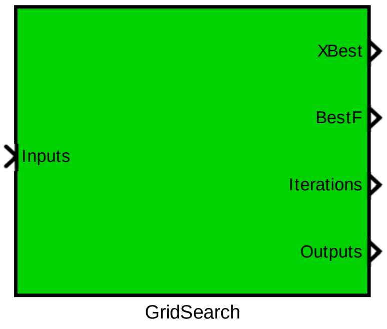
Figure 2: The Grid Search Method applied on the White Mountain function: ● - evaluation grid, ● - centroid, ⋆ - optimum.

```
XBest =

   1.0e+06 *

     0.3161    4.9043

BestF =

  6.2800e+03

Iters =

   199

Elapsed time is 0.218983 seconds.
```

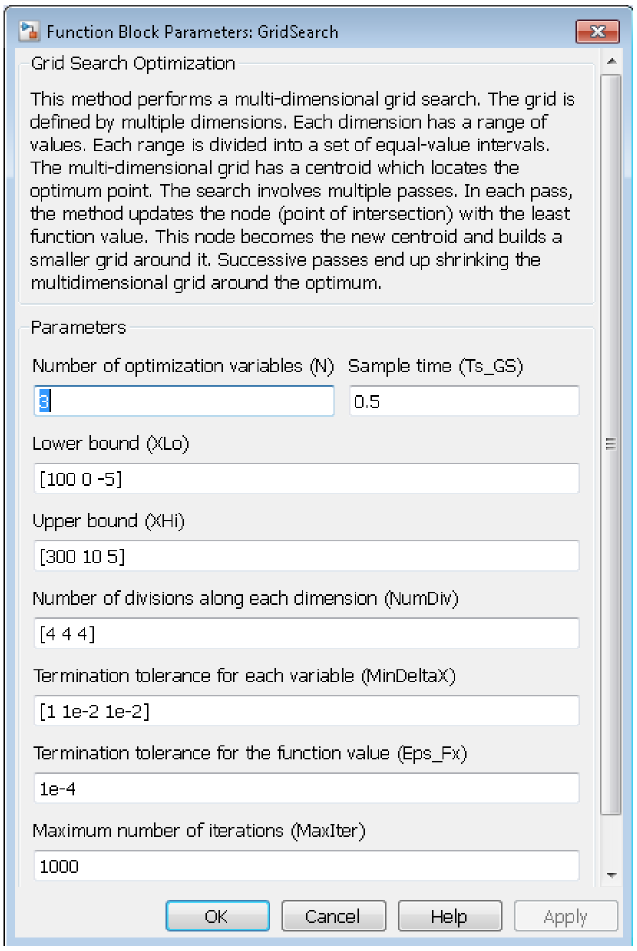
The surfaces used to test the algorithms are highly nonlinear and have a non-smooth behavior. In addition, the surfaces are characterized by local minima and maxima. These surfaces are very challenging for gradient based solvers that can easily get stuck in a local minima or maxima. The Grid Search Method is a gradient free approach. From the results we see that the Grid Search Method successfully finds the global solution with reasonable computational complexity.

3 Simulink implementation

This section presents the Matlab/Simulink implementation of the Grid Search Method. Two versions of the method are implemented: with constant bounds and with variable bounds. For brevity, only the one with constant bounds is presented here. In Fig. 3, we illustrate the Grid Search Simulink block and the interface that allows user to specify the optimization parameters. The Grid Search Simulink block consists of two iterators. The function or Simulink block to be optimized needs to be placed inside the second iterator (see Appendix).



(a)



Grid Search Optimization

This method performs a multi-dimensional grid search. The grid is defined by multiple dimensions. Each dimension has a range of values. Each range is divided into a set of equal-value intervals. The multi-dimensional grid has a centroid which locates the optimum point. The search involves multiple passes. In each pass, the method updates the node (point of intersection) with the least function value. This node becomes the new centroid and builds a smaller grid around it. Successive passes end up shrinking the multidimensional grid around the optimum.

Parameters

Number of optimization variables (N): 3

Sample time (Ts_GS): 0.5

Lower bound (XLo): [100 0 -5]

Upper bound (XHi): [300 10 5]

Number of divisions along each dimension (NumDiv): [4 4 4]

Termination tolerance for each variable (MinDeltaX): [1 1e-2 1e-2]

Termination tolerance for the function value (Eps_Fx): 1e-4

Maximum number of iterations (MaxIter): 1000

(b)

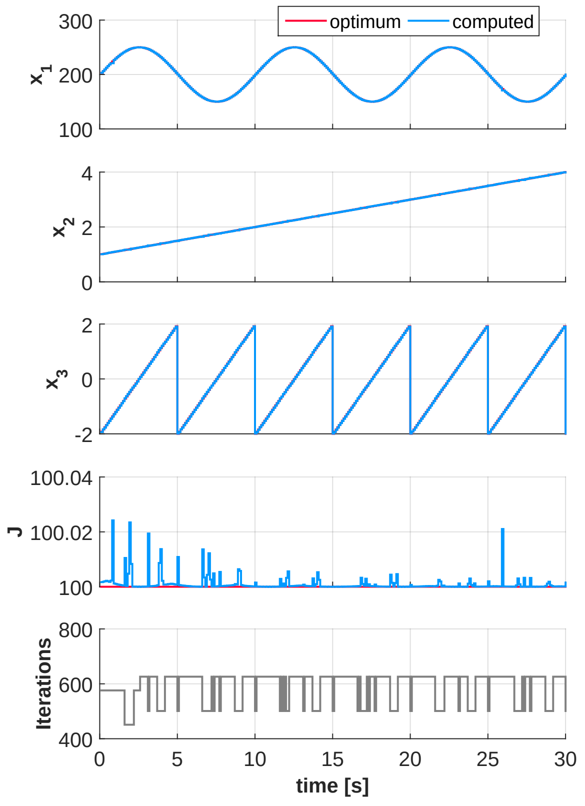
Figure 3: (a) The Grid Search Simulink block. (b) The Grid Search Simulink block parameters.

Example

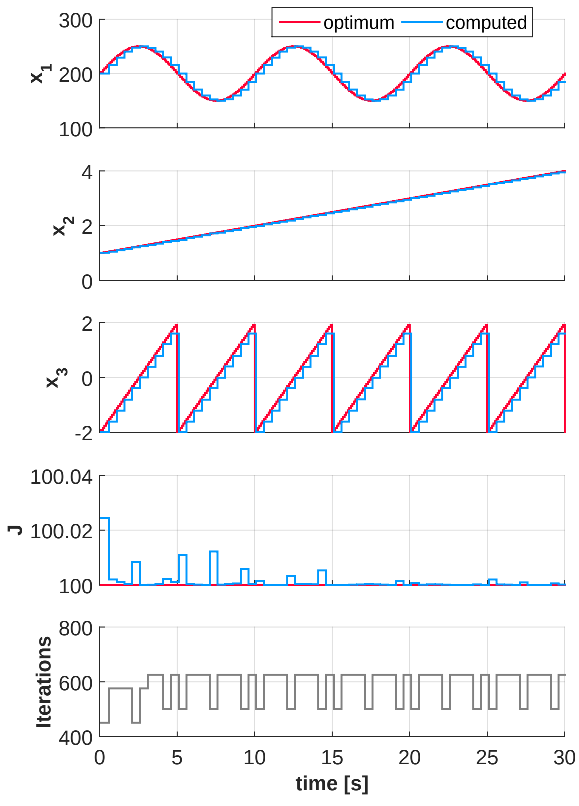
In what follow we illustrate the utility of using this method to track online a certain optimum. To exemplify, we define a nonlinear 3 input objective function as:

$$J(x, x_{opt}) = J_{min}\left(1 + \frac{1}{x_{1opt}}(x_1 - x_{1opt})^2 + \frac{1}{x_{2opt}}(x_2 - x_{2opt})^2 + (x_3 - x_{3opt})^4\right) \quad (2)$$

where, $x = [x_1\ x_2\ x_3]^\top$ is the input to be computed and $x_{opt} = [x_{1opt}\ x_{2opt}\ x_{3opt}]^\top$ is the optimum to be found. For exemplification we choose here $J_{min} = 100$. Note that by knowing the function it is easy to see that the optimum of $J = J_{min}$ is reached when $x = x_{opt}$. In what follows, we check the effectiveness of the Grid Search Method by applying it to function (2).



(a)



(b)

Figure 4: (a) Grid Search Method results computed at a sampling rate $T_s = 0.1$ s. (b) Grid Search Method results computed at a sampling rate $T_s = 0.5$ s.

In Fig. 4, the simulation results are illustrated. Two simulations were performed, one in which the algorithm is triggered at the base sampling rate and one in which the algorithm is triggered at slower sampling rate as compared to the base sampling rate. From the simulation results, it can be seen that the method successfully finds the three optimum values. Around 600 function evaluations are needed in order to find the optimum with an accuracy of 10^{-4}.

Remark 1 *Due to the fact that Simulink does not accept any block needing elapsed time (such as Discrete-Time Integrator block) placed within the While Iterator subsystem, at this moment the Grid Search Simulink block cannot be used to optimize a dynamic system. This will require a system solver within the Simulink solver, which is not available in the latest Matlab release (R2014b). A work around is to compute analytically the solution for the dynamic system, such that the While Iterator subsystem sees it as a static block.*

4 Conclusions

In this report the Grid Search Method has been presented. The method has been shown to be effective in finding the global solution for an optimization problem that is non-smooth, nonlinear and is characterized by local minima or maxima. The method can be used online and offline to solve multidimensional problems.

To reduce computational complexity while increasing the accuracy, the Grid Search Method can be combined with a gradient-based method. First, specify a less tight tolerances for the Grid Search Method and find a solution close to the global optimum. Second, trigger a gradient-based method to obtain the optimum solution at the required accuracy.

References

[1] Namir Shammas, *Multivariable optimization by grid search*, http://www.namirshammas.com/MATLAB/mainMATL.htm

5 Appendix

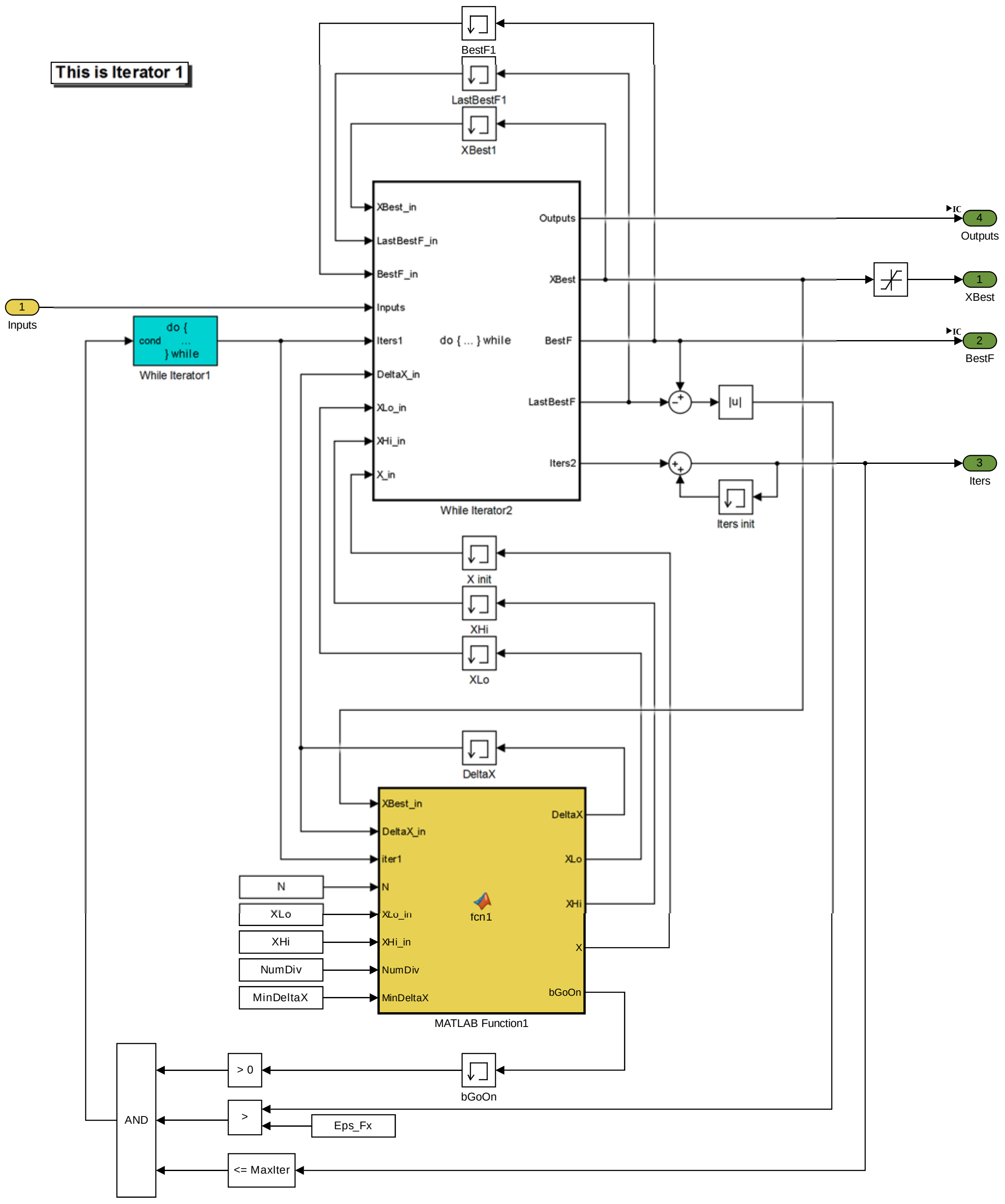


Figure 5: Iterator 1 of the Grid Search Method.

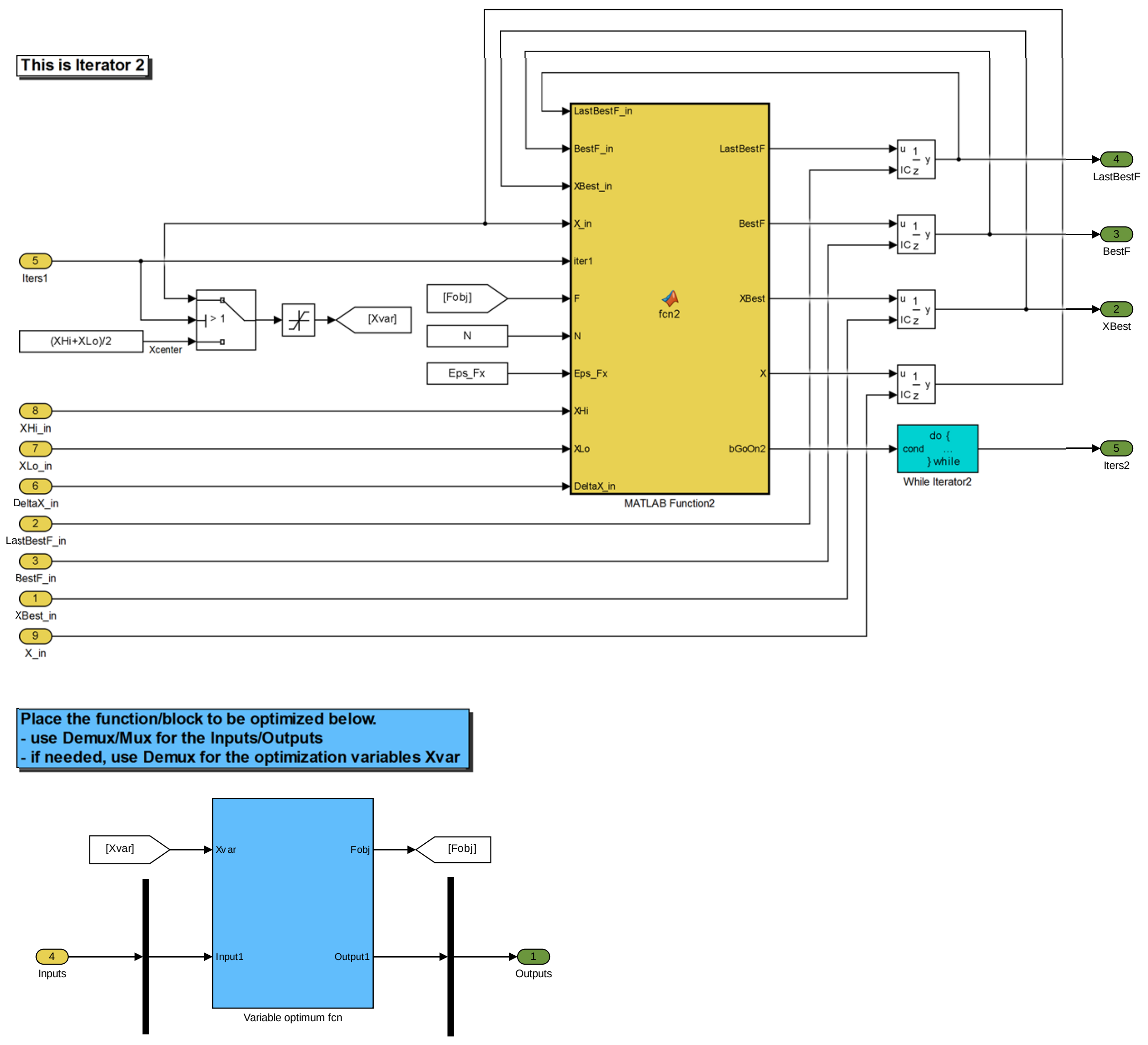


Figure 6: Iterator 2 of the Grid Search Method.